Problem 1 .

1. Given is a Truth Table

full adder

X	Y	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

1. Realize this function using BDDs. Arbitrary order of variables

2. Realize this function using Kronecker Functional Decision Diagrams. Arbitrary order of variables.

3. Show solution based on mapping BDD or KFDD to arbitrary logic gates. Draw a schematic and write expressions.

4. Realize this function using Feynman and Toffoli gates. Show and explain your design stages. Verify your solution.

In each of the above 3 sub-problems, try to minimize the total number of gates and verify your solutions.

full adder

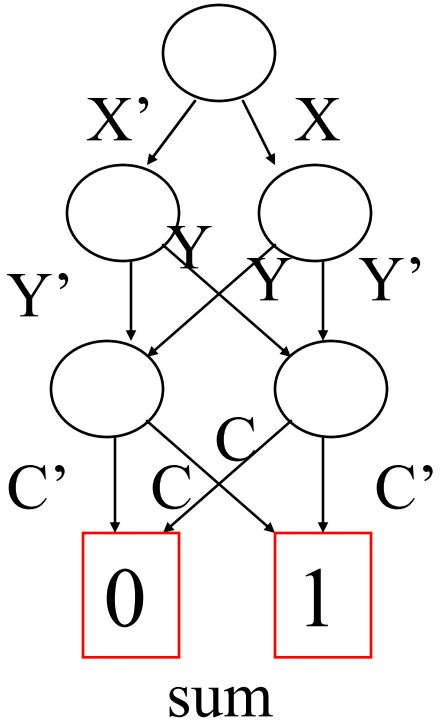
X	Y	Cin	Sum	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

XY \ C	0	1
00	0	1
01	1	0
11	0	1
10	1	0

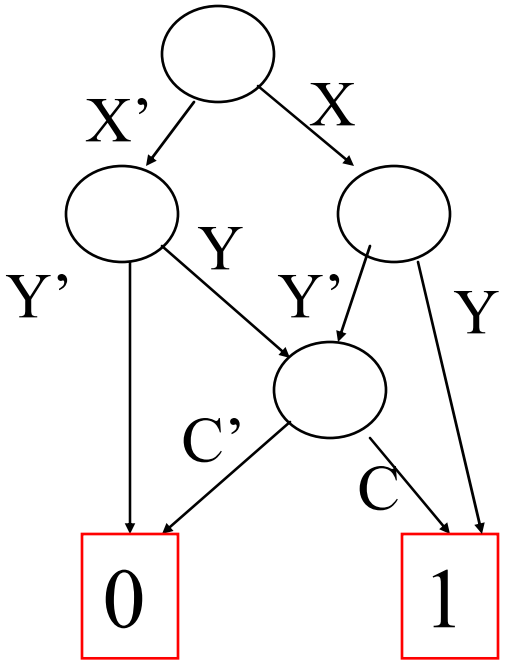
sum

XY \ C	0	1
00	0	0
01	0	1
11	1	1
10	0	1

Carry out



sum



carry

Binary Decision Diagrams

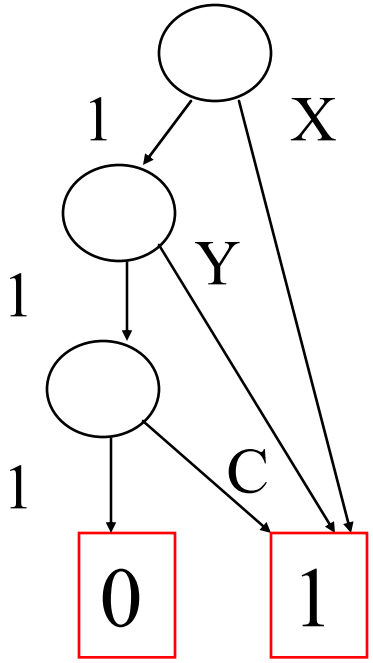
Realize this function using Kronecker Functional Decision Diagrams. Arbitrary order of variables.

XY \ C	0	1
00	0	1
01	1	0
11	0	1
10	1	0

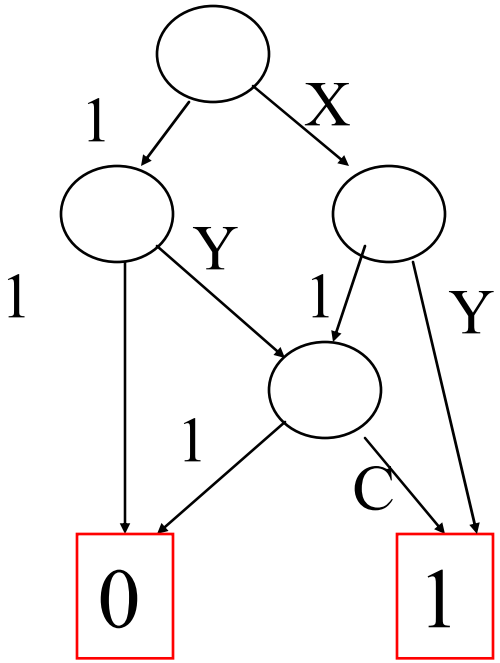
sum

XY \ C	0	1
00	0	0
01	0	1
11	1	1
10	0	1

Carry out



sum

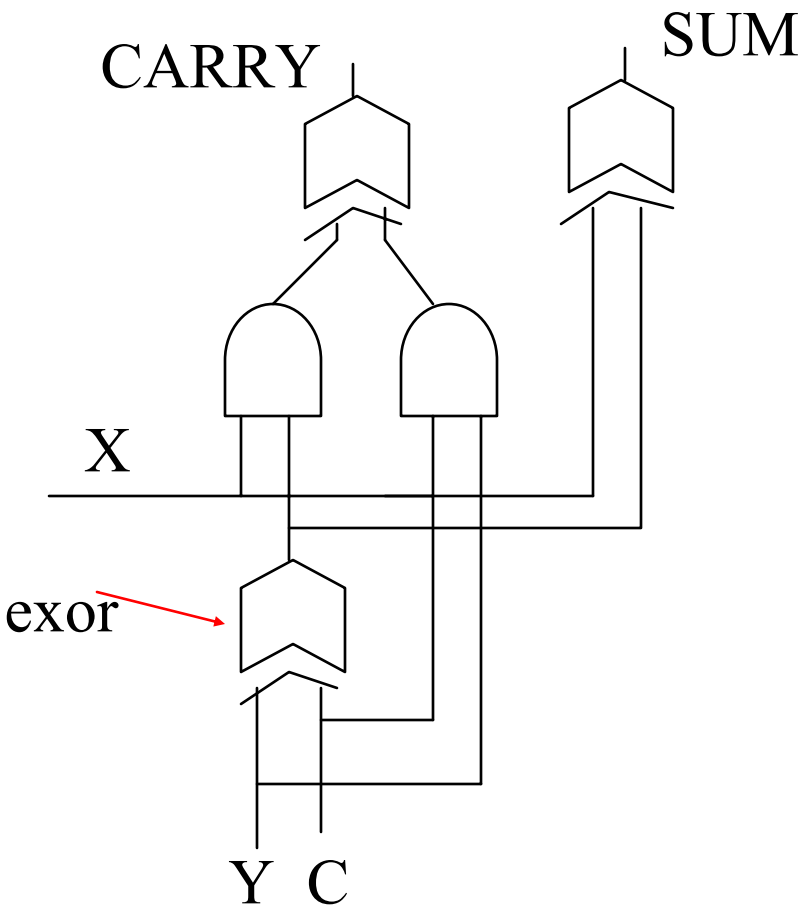


carry

3. Show solution based on mapping BDD or KFDD to arbitrary logic gates. Draw a schematic and write expressions.

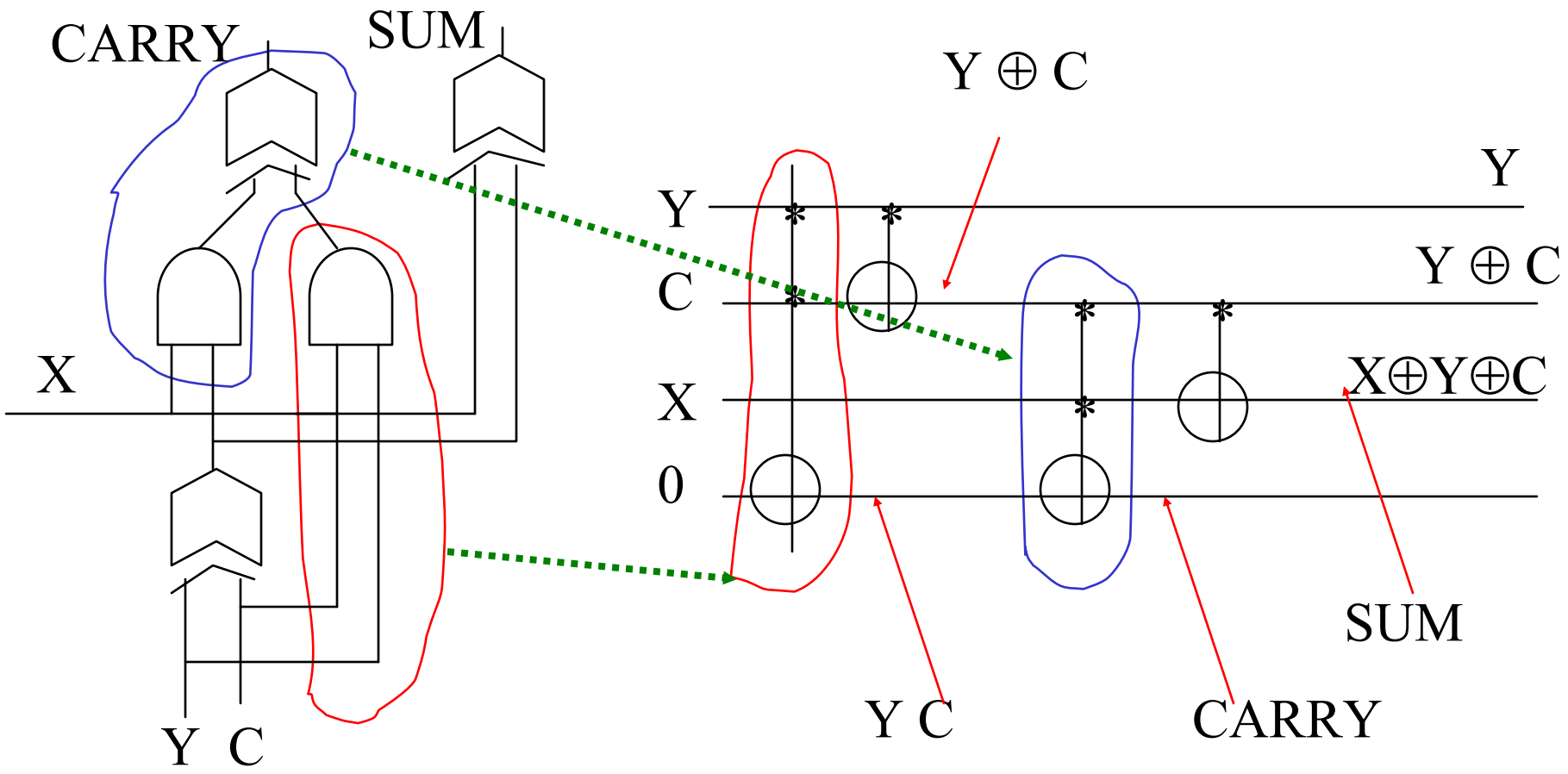
Replacing nodes of BDD with multiplexers we obtain a well known realization of parity or exor sum of variables circuits

Another method to observe the expressions which leads to SUM = X⊕ Y ⊕ C realized with two EXOR gates, and CARRY = XY ⊕ XC ⊕ YC = X(Y ⊕ C)⊕ YC



Realize this function using Feynman and Toffoli gates.

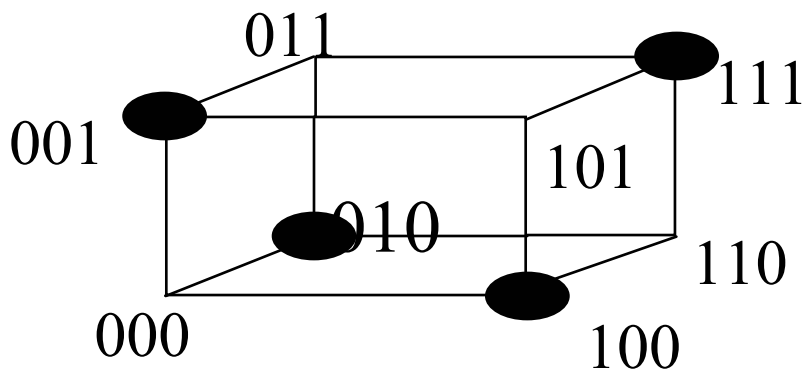
Show and explain your design stages.



As this figure shows, a good multi-level circuit from 2-input AND and EXOR gates can be often rewritten to a reversible circuit with approximately the same number of gates as there are EXOR gates in the first circuit. AND/EXOR is replaced with Toffoli and EXOR with Feynman gates. Sometimes Feynman gates are added for copying arguments.

Problem 2.

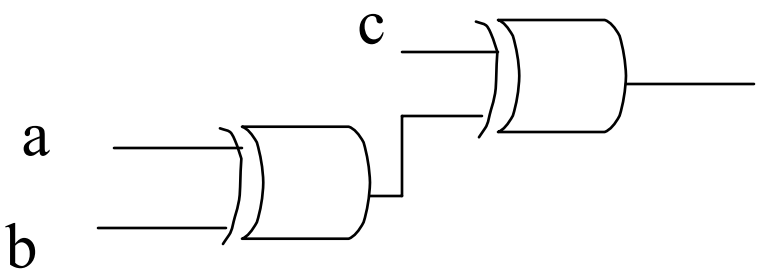
Realize this function of 3 variables as a circuit with MINIMUM number of NAND gates, each with 2 inputs. Try to prove that your solution is minimum.



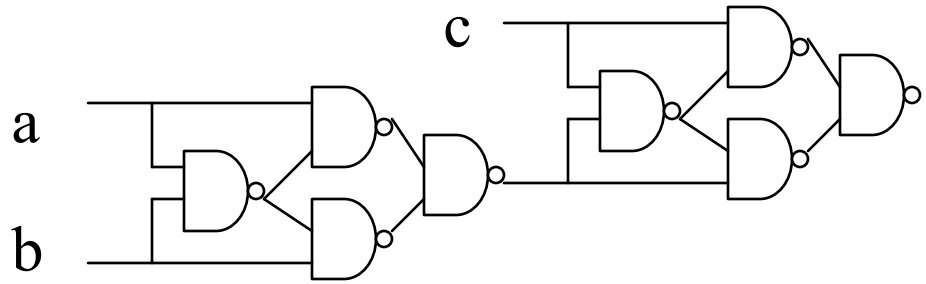
- We draw a Kmap.

a \ b c	00	01	11	10
0	0	1	0	1
1	1	0	1	0

We recognize a “Chess-Pattern” typical for linear (EXOR only) circuits.



Now we synthesize the EXOR gate using 2-input NAND gates and we replace EXOR gates with their NAND realizations (macro-generation).

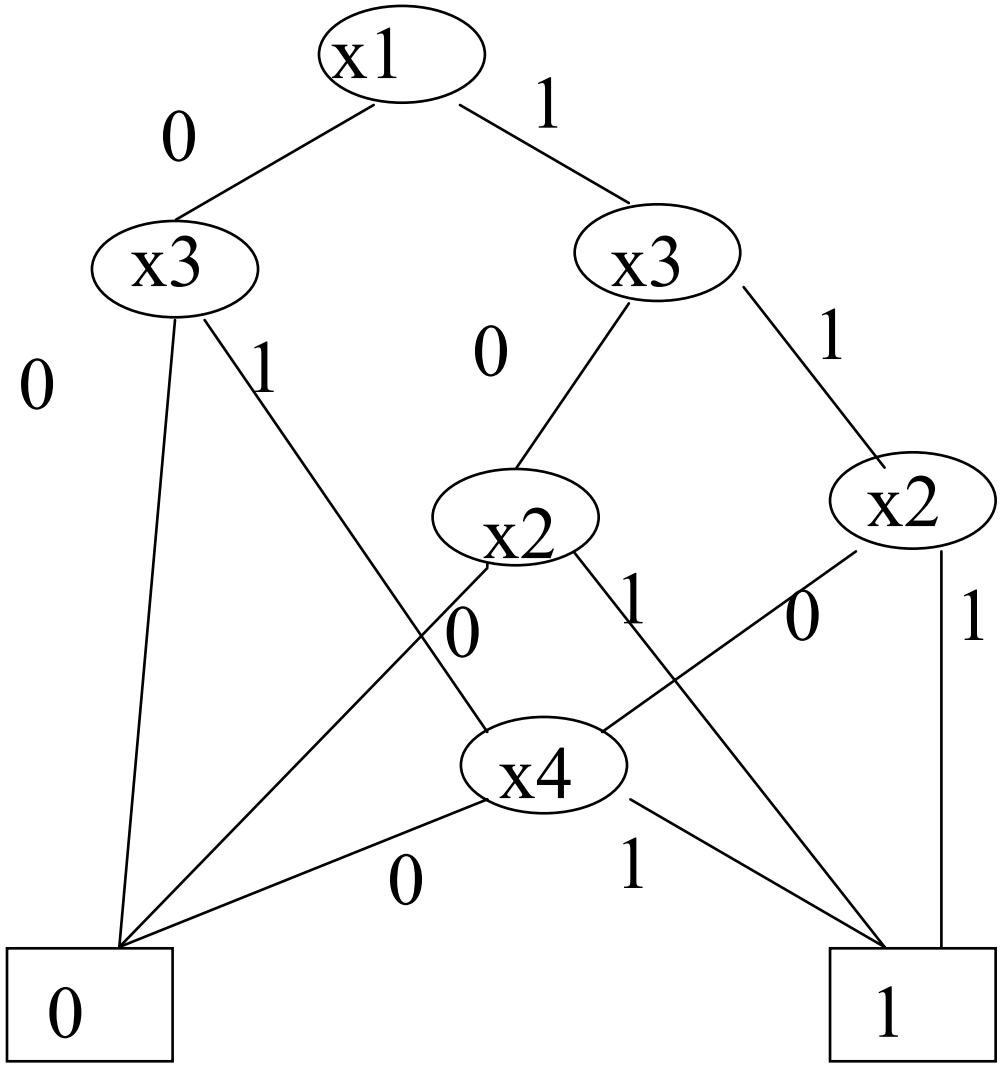


Since this is a linear circuit, there is no better decomposition to 2-input gates than to EXORs. Moreover the decomposition is disjoint. This is not a formal proof, but it is very unlikely that a better solution exist to this problem.

Problem 3

Find a better ordering for this function or show that it does not exist.

Realize the function from the improved BDD using only majority gates (majority gate is f=ab+ac+bc). Minimize the number of majority gates. Verify the solution using Kmap.



Find a better ordering for this function or show that it does not exist.

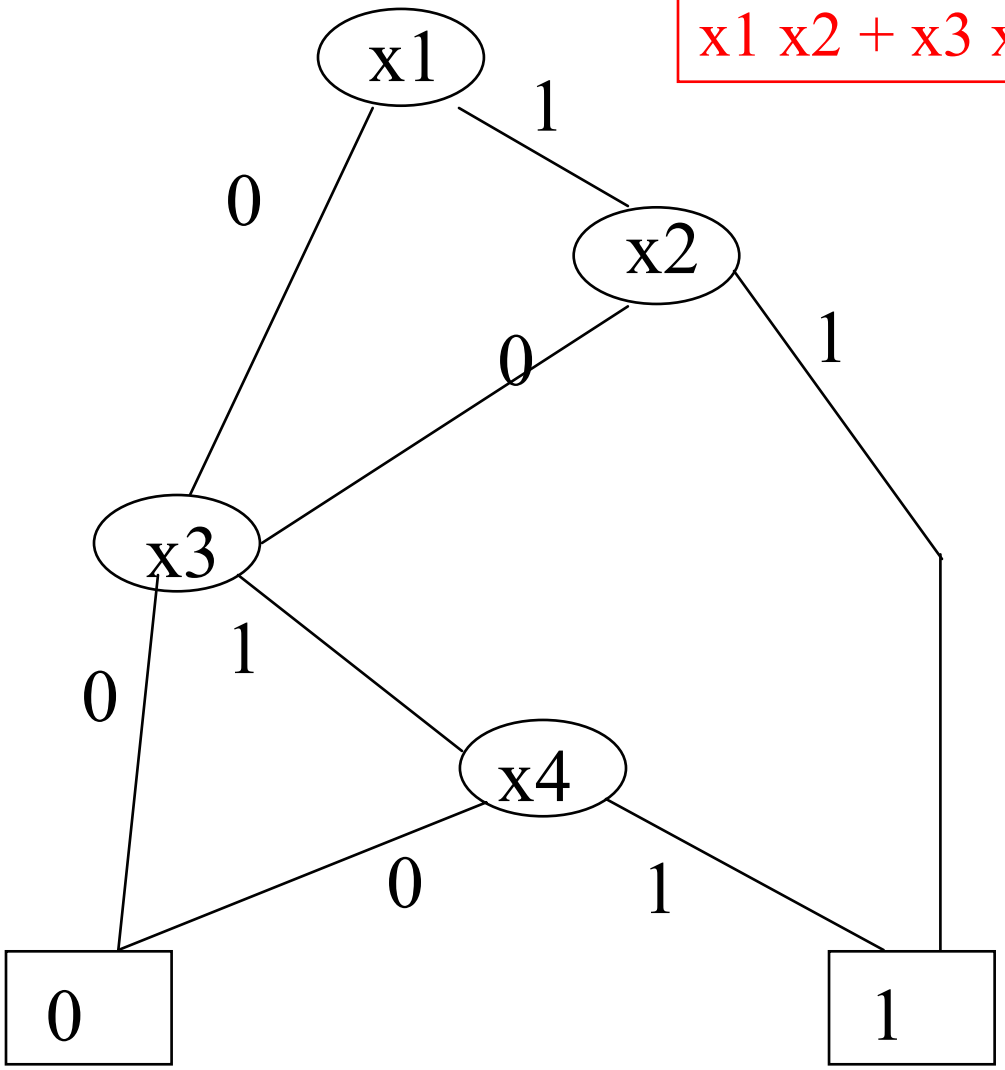
As always, first we have to draw the Kmap to understand what kind of Boolean function we are dealing with here.

By analyzing paths from root note do node 1 the function is x1 x3 x2 + x1 x3’ x2 + x1’x3 x4 + x1x3 x2’ x4 = x1 x2 + x3 x4

x1x2 \ x3x4	00	01	11	10
00	0	0	1	0
01	0	0	1	0
11	1	1	1	1
10	0	0	1	0

As we see, order of variables in BDD x1 x3 x2 x4 was not creating minimum SOP because it was splitting , for instance, Prime Implicant x1 x2 to two implicants. Thus the order should be x1x2 x3 x4. In essence, the variable x1 and x2 can be in any order before variables x3 x4 in any order.

Let us draw BDD with order x1x2x3x4



• In some problems we assume that input negations exist in some other we assume that they do not exist, are not available to the designer.

•Here you are requested to use only majority functions.

•So we cannot create a negation using majority.

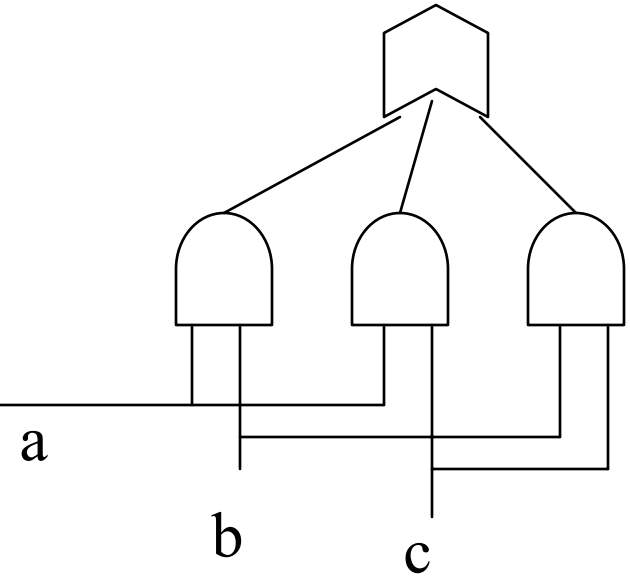
•Thus, to design arbitrary function with majority gates you have to assume that input negations are available

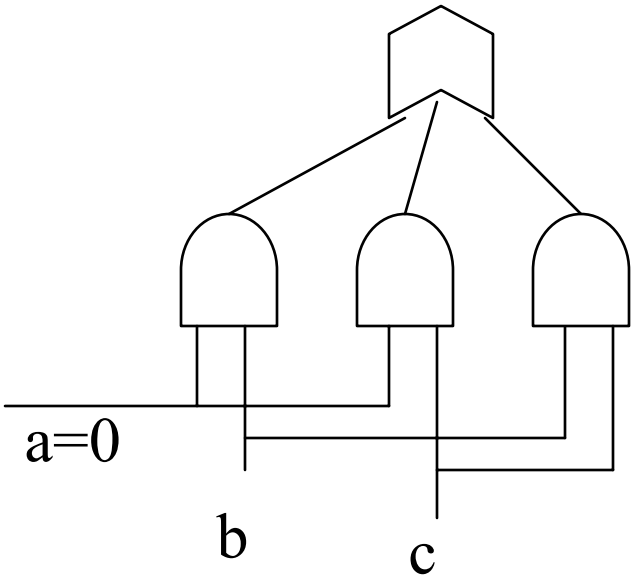
• The only useful cofactors of majority are two-input AND and OR gates

•We can build from them.

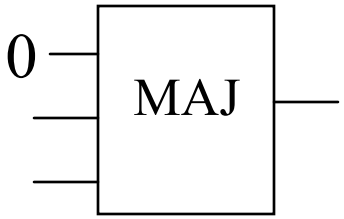
•In our function, which is positive unate, we do not need negated arguments.

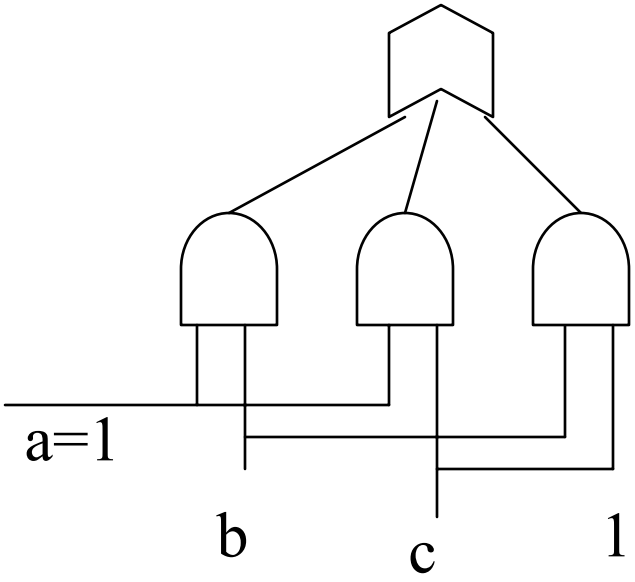
•See next slide..



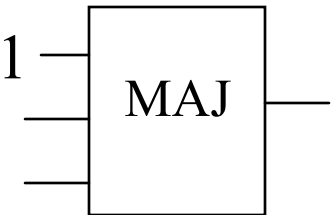


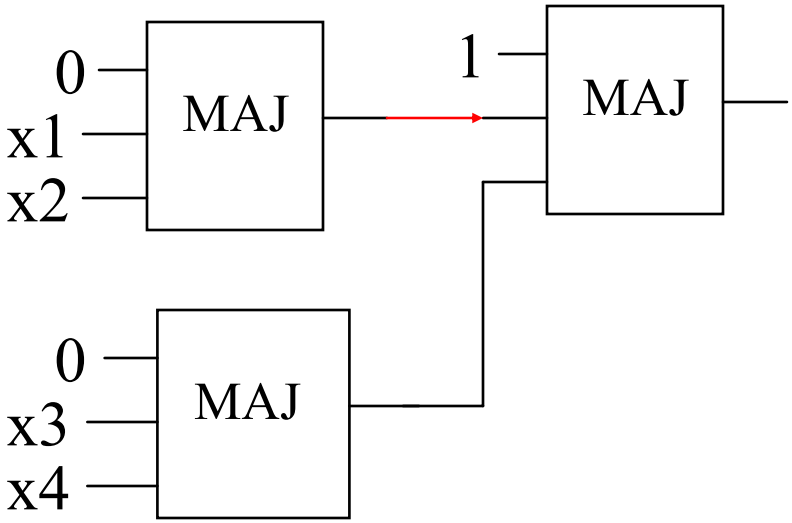
Realization of AND gate bc with majority





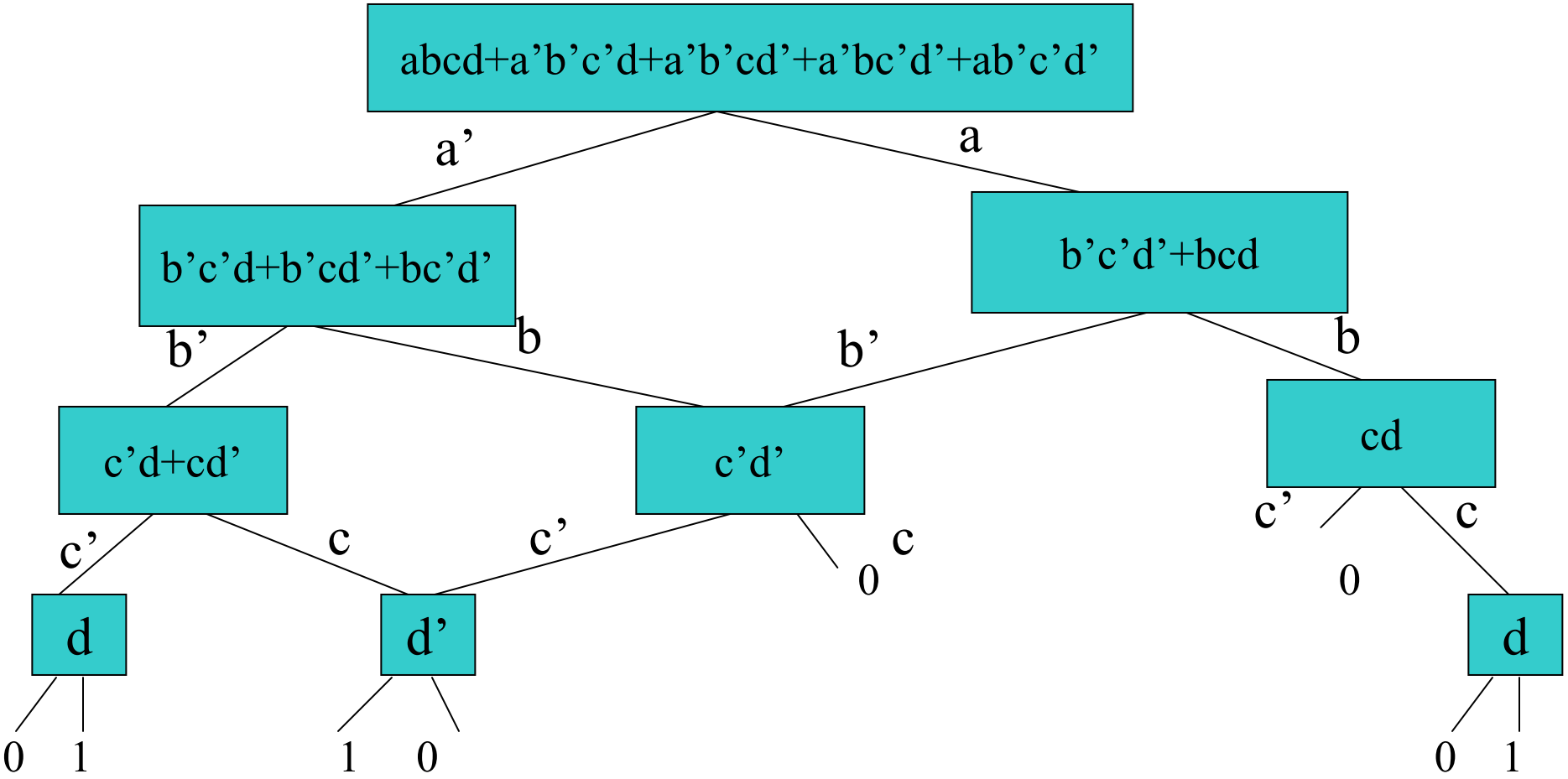
Realization of OR gate b+c with majority



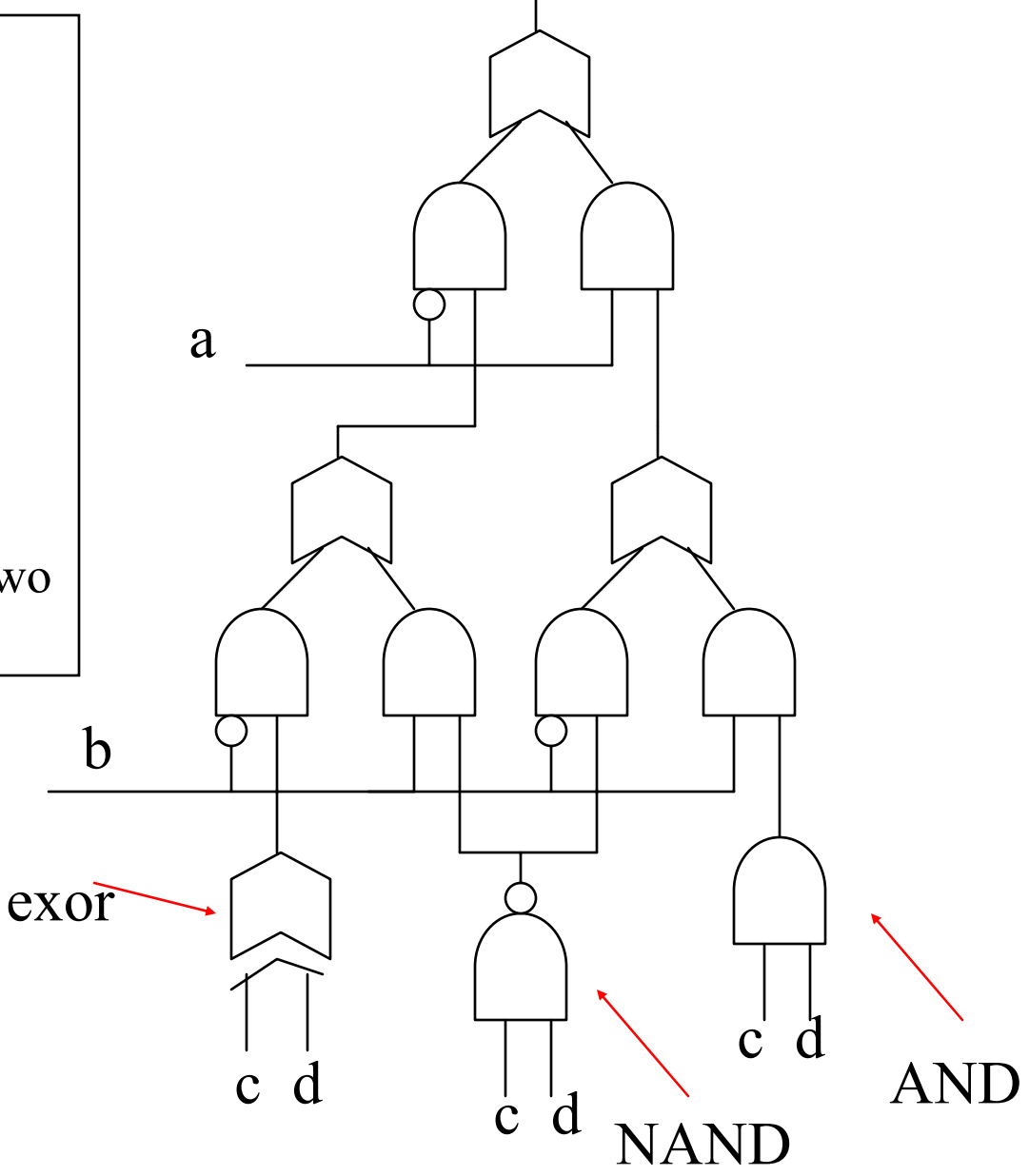


Problem 4

Realize the function expressed by this lattice diagram using only 2-input gates (arbitrary). Minimize the number of used gates. Draw a Kmap of this function and verify your solution.



- First we see that this is a symmetric function $S^{1,4}(a,b,c,d)$.
- Thus our circuit is a Shannon Lattice.
- Every Shannon Lattice can be realized with mulitplexers only.
- Next we can transform multiplexers to other gates with two inputs.



ab \ cd	00	01	11	10
00	0	1	0	1
01	1	0	0	0
11	0	0	1	0
10	1	0	0	0

$\frac{\partial f}{\partial x}$ is called Boolean difference of f

with respect to x

Problem 5

$$\frac{\partial f}{\partial x} = f_x \oplus f_{\overline{x}}$$

f is sensitive to the value of x when $\frac{\partial f}{\partial x}$ *is not* 0

Find two practical applications of the Boolean Difference operation defined as above. Think about test generation and the condition for an input combination to be a test.

How to realize this operation using Cube Notation? (Cube is a product of variables and negated variables).

How to realize it using Truth Tables or Kmaps?

How to realize it using BDDs?

How to realize it using Functional Decision Diagrams (FDDs)?

Find two practical applications of the Boolean Difference operation defined as above. Think about test generation and the condition for an input combination to be a test.

1. Function F depends on variable x if its Boolean Difference with respect to x is not zero. Then we can check every variable x of a completely specified function and these variables for which the derivative is zero can be removed from the set of function variables. Thus the number of arguments is smaller in some cases.

2. If Boolean Difference of Function F for some signal x in a network (with one of its outputs = F) with respect to x is one, then the change in this signal propagates to the output F. Thus a condition to find all tests for stuck-at-zero fault in signal x is

x * df/dx ⟵ Every minterm that satisfies this formula is a test. Variable x=1 since it must be opposite to stuck-at-fault value.

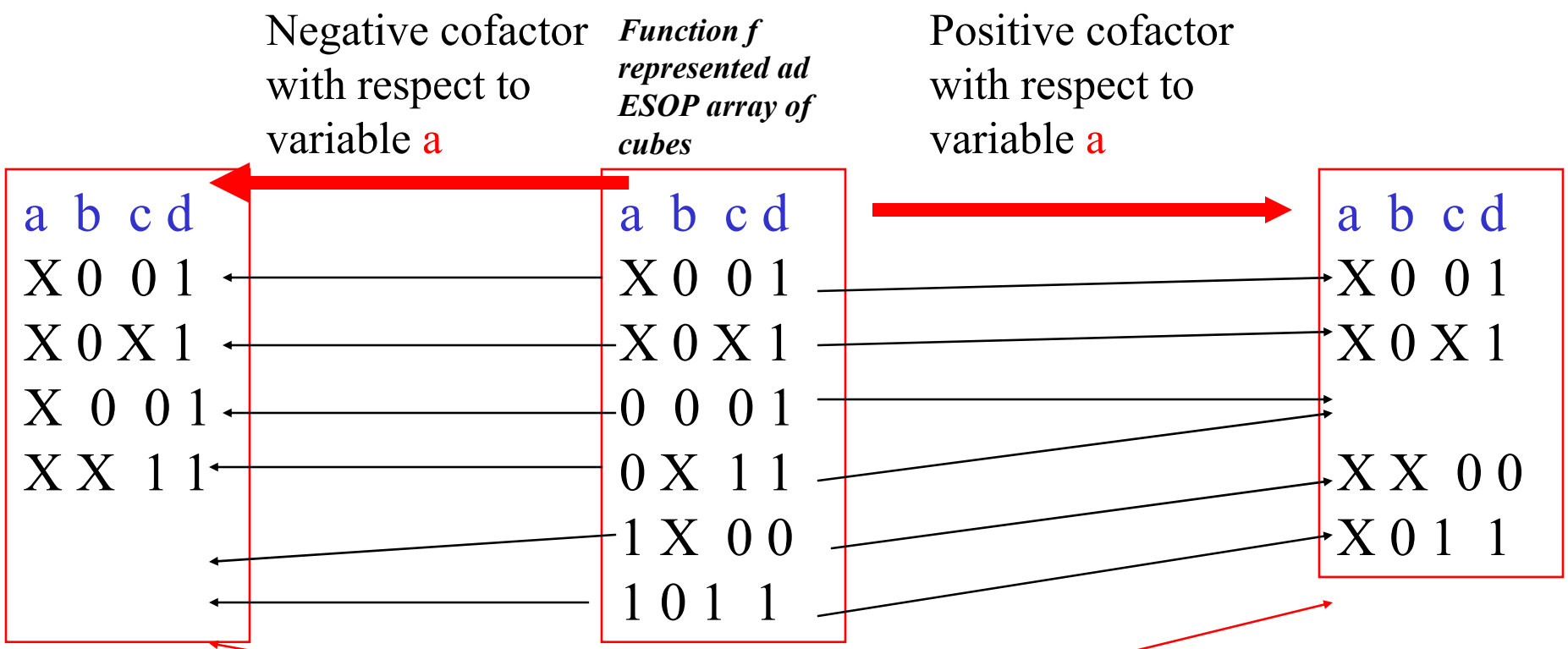
Similarly a condition to find all tests for stuck-at-zero fault in signal x is

x' * df/dx ⟵ Every minterm that satisfies this formula is a test for stuck-at-one fault in signal x

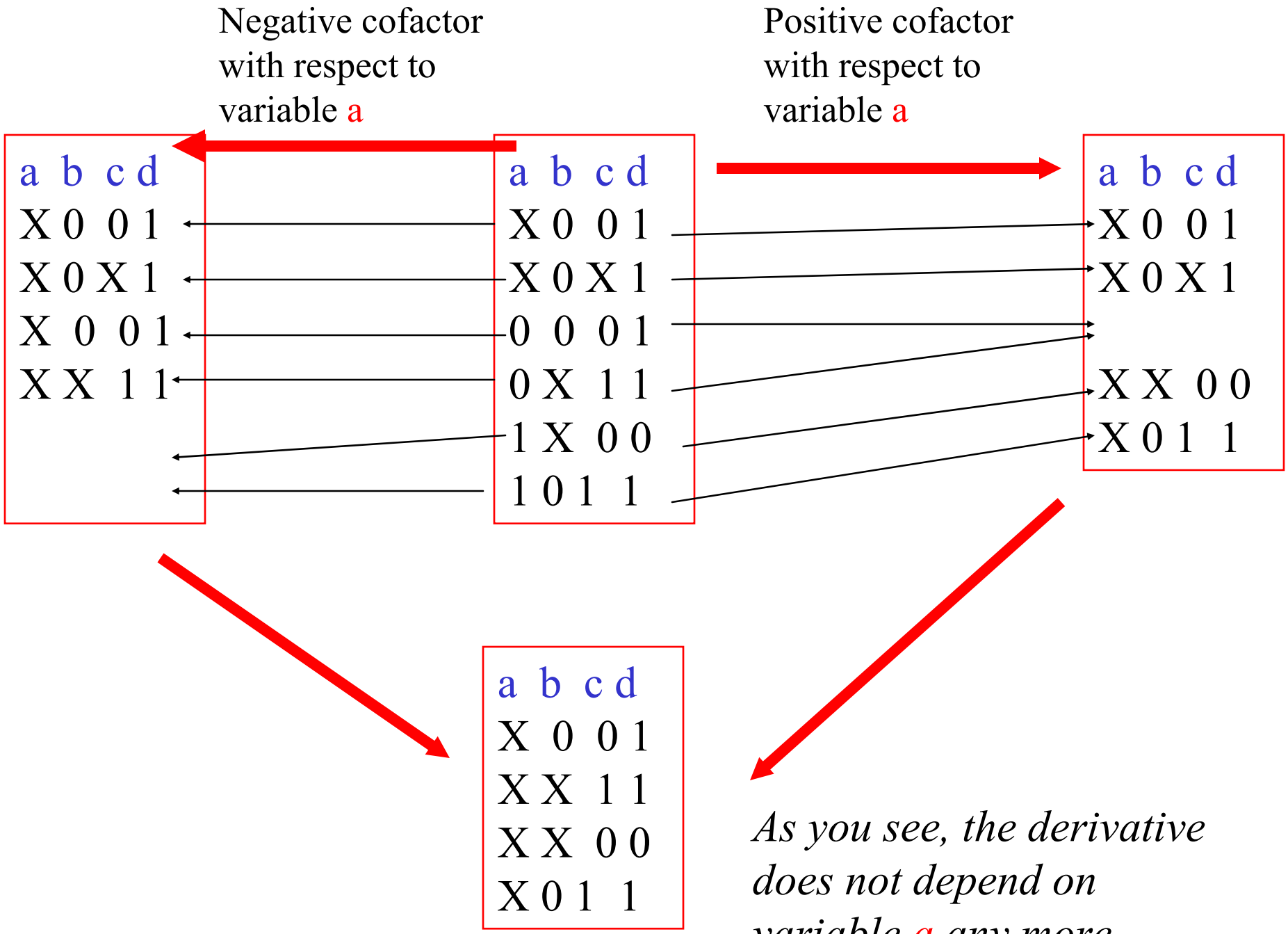
Variable x=0 since it must be opposite to stuck-at-one value. Thus x'=1.

How to realize this operation using Cube Notation? (Cube is a product of variables and negated variables).

Cube is a product of literals. Thus positive cofactor of a cube that includes literal x is the product without this literal. If the cube does not include this literal, the whole cube is omitted. Similarly negative cofactor is calculated. Example



Now we perform union of these two arrays of cofactors and we remove the repeated cubes.


Negative cofactor with respect to variable a
Positive cofactor with respect to variable a
a b c d
X 0 0 1
X 0 X 1
X 0 0 1
X X 1 1
a b c d
X 0 0 1
X 0 X 1
0 0 0 1
0 X 1 1
1 X 0 0
1 0 1 1
a b c d
X 0 0 1
X 0 X 1
X X 0 0
X 0 1 1
a b c d
X 0 0 1
X X 1 1
X X 0 0
X 0 1 1
As you see, the derivative does not depend on variable a any more

How to realize it using Truth Tables or Kmaps?

We use Kmap folding method. Graphically perform EXOR of mirror cells of the map. This is SOP, another function.

a	b	c	d
X	0	0	1
0	X	1	X
1	1	X	X

ab \ cd	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11	1	1	1	1
10	0	1	0	0

Perform in each cell of the maps exor operator on values 0 and 1

ab \ cd	00	01	11	10
00	0	1	1	1
01	0	0	1	1
11				
10				

$\oplus$

ab \ cd	00	01	11	10
10	0	1	0	0
11	1	1	1	1
01				
00				

$=$

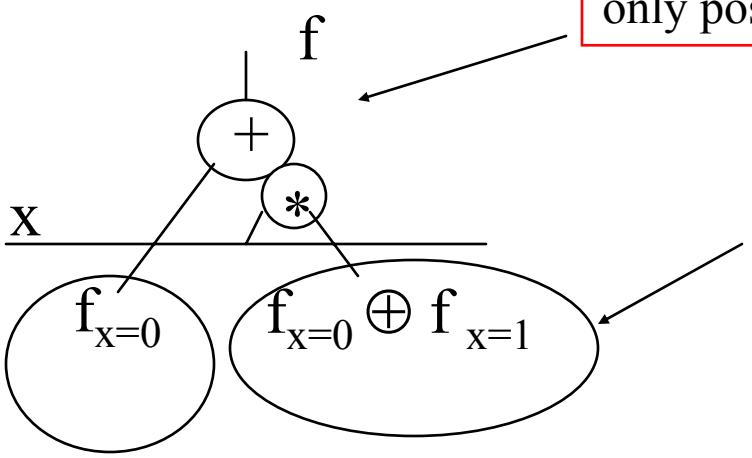
ab \ cd	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11				
10				

How to realize it using BDDs?

Create a BDD with the variable that you want to differentiate for at the top. Now calculating of the negative and the positive cofactors is easy. They are two BDDs that are pointed by 0-edge and 1-edge of the top node in the BDD, respectively. Next do the EXOR of two cofactors. Use APPLY operator on BDDs.

How to realize it using Functional Decision Diagrams (FDDs)?

Positive Davio Node, as we know only positive davio nodes are in FDDs



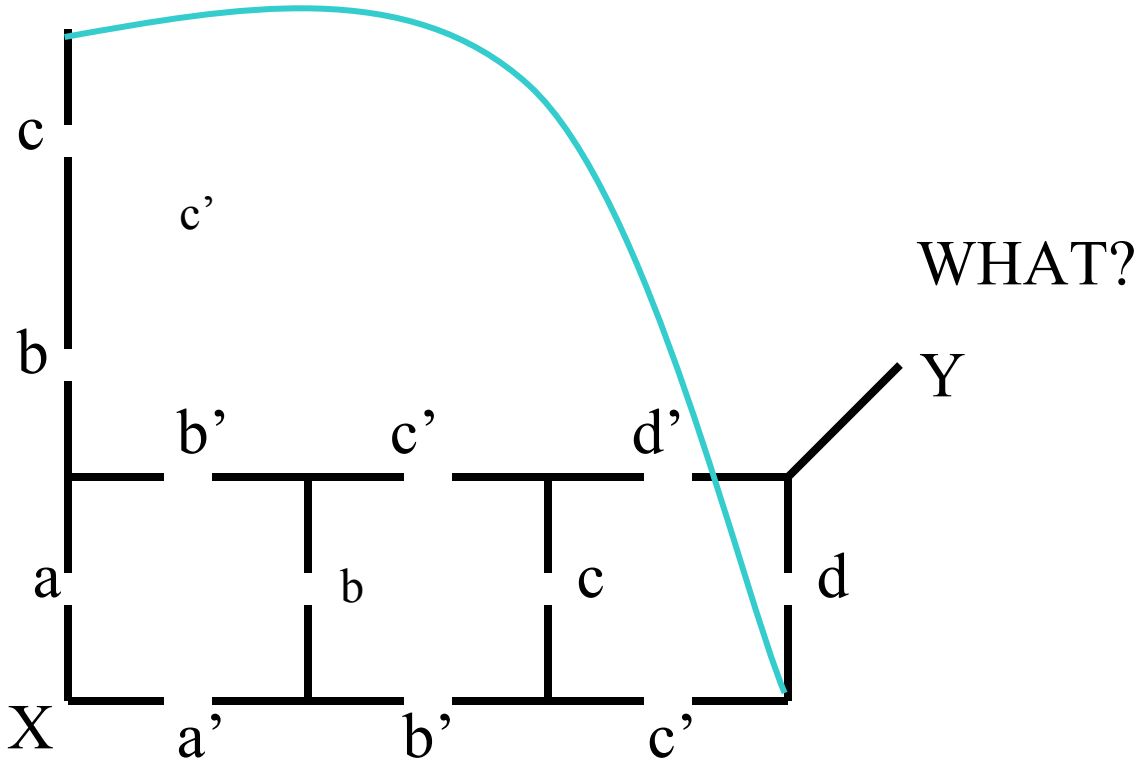
Variable on top of FDD

This node is a Boolean difference with respect to variable x that is on top of the FDD.

PROBLEM 6

1. What is the function realized here? Show Kmap and analyze all paths.

2. Realize this function with only multiplexers.



By analyzing all paths from point X to point Y we see that this is a symmetric function $S^{1,4}(a,b,c,d)$

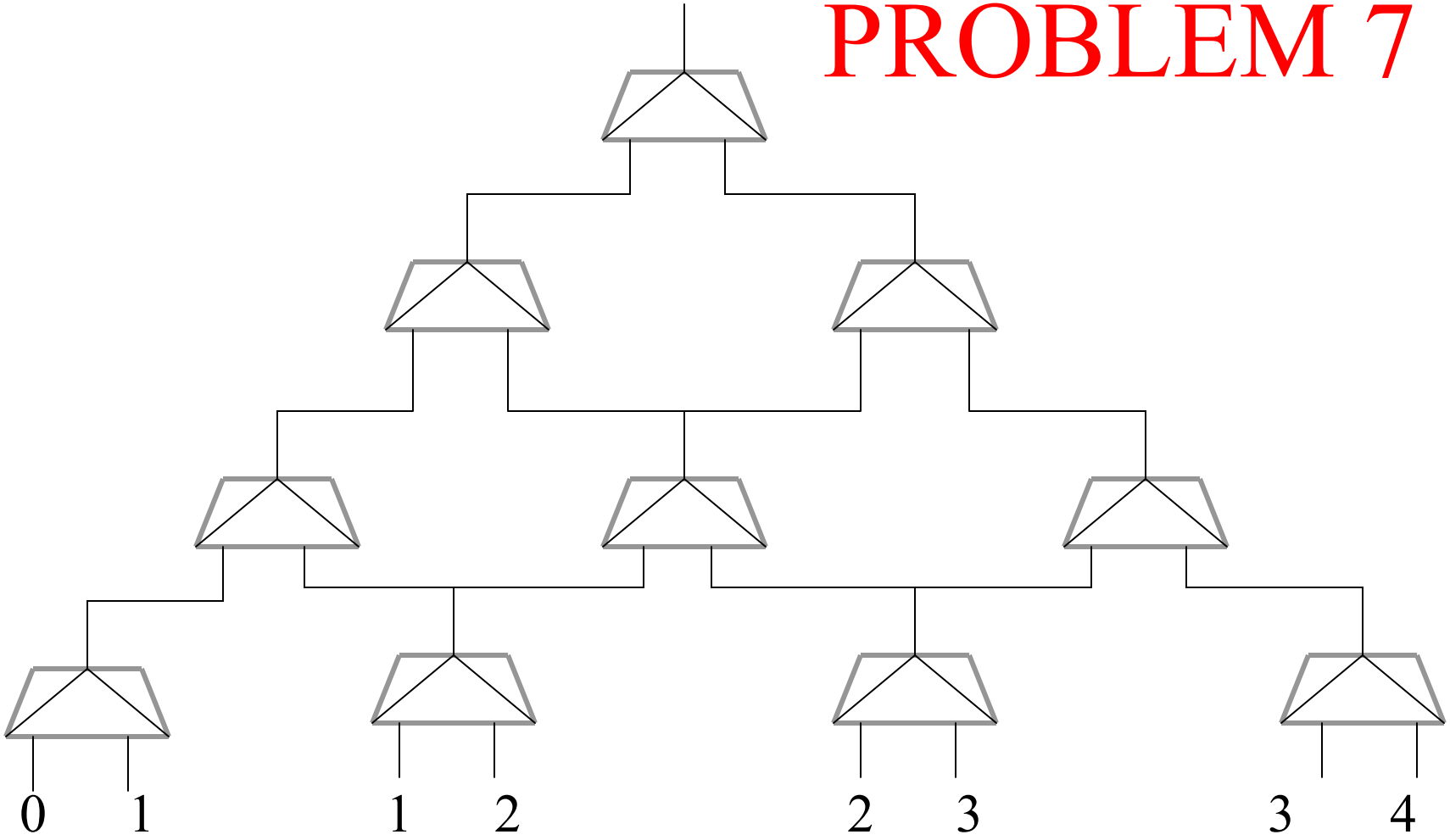
• Symmetric function $S^{1,4}(a,b,c,d)$ can be build either using Shannon Lattice of a BDD. Both structures use only multiplexers to realize Shannon nodes.

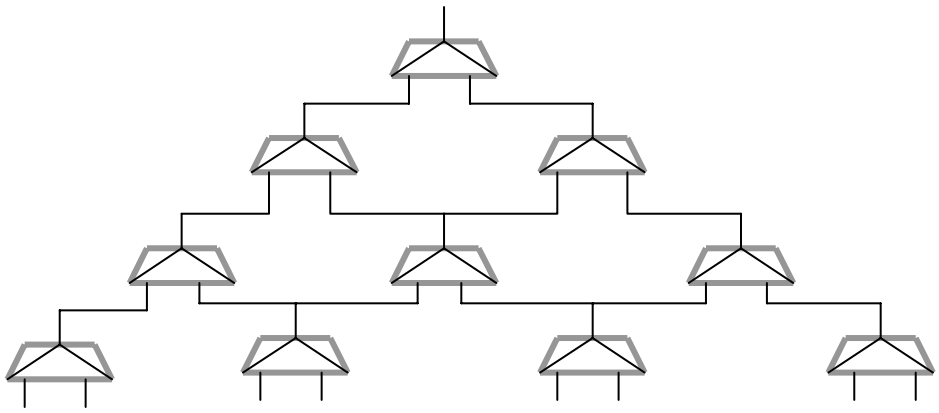
• Observe that this is the same function as in problem 4.

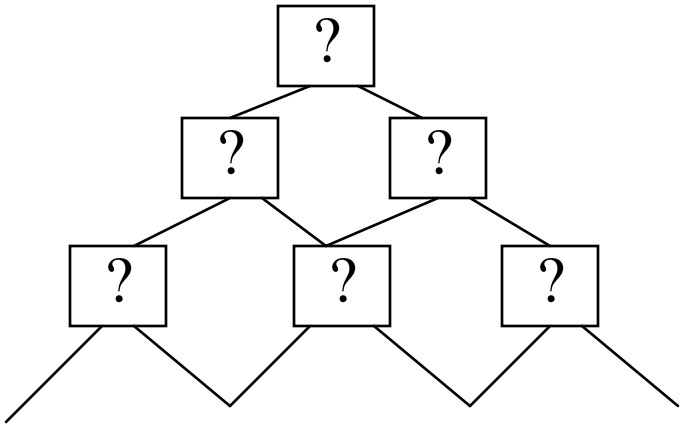
• So, a student who has enough experience with diagrams and symmetric functions, can solve problems 4 and 6 very quickly.

The symmetric function can be mapped in to the lattice of multiplexers as shown in the figure below. Can you prove that by replacing multiplexers with ARBITRARY other gates we will be able still to realize arbitrary symmetric function. Control variables of multiplexers are not shown here.

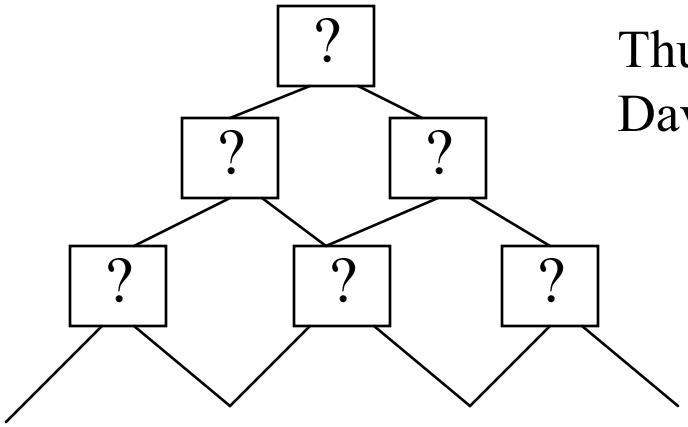
PROBLEM 7



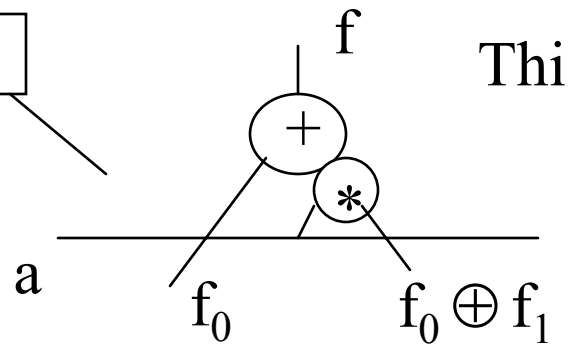




The statement in the problem formulation says " replacing multiplexers with ARBITRARY other gates" which means that we have a choice of gates. If we will be able to find some gate which will be playing the role equivalent to multiplexer in Shannon Lattice, it will be ok. But we know from the class that there are three expansions for a single variable – Shannon, Positive Davio and Negative Davio. They all reduce the variable. Then we can select, instead of multiplexer that corresponds to Shannon a Positive Davio gate that corresponds to Positive Davio expansion. Now we have to prove that every symmetric function is realized in a rectangular (lattice) structure of gates as above. This is called Davio Lattice or Reed-Muller Lattice and was discussed in Homework 2 by some of you, also it was mentioned and analyzed in class.



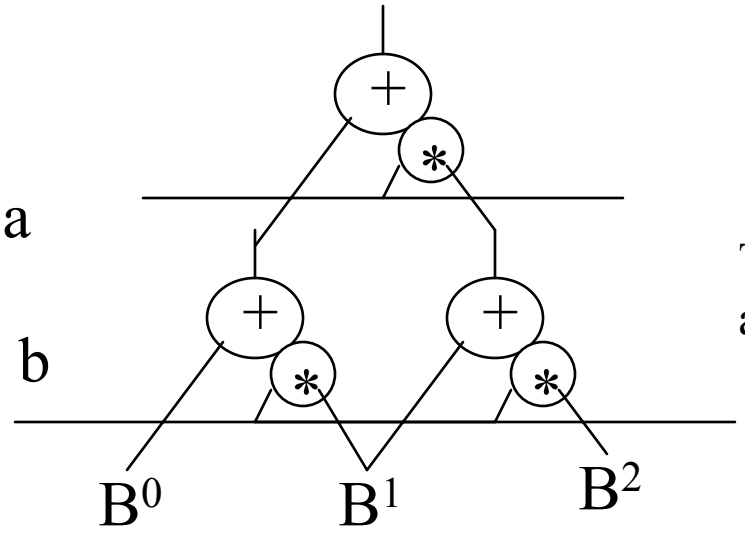
Thus let us replace the question mark by Positive Davio gate and see what happens



This is Positive Davio gate

$$f = a' f_0 \oplus a f_1 = (1 \oplus a) f_0 \oplus a f_1 = f_0 \oplus a (f_0 \oplus f_1)$$

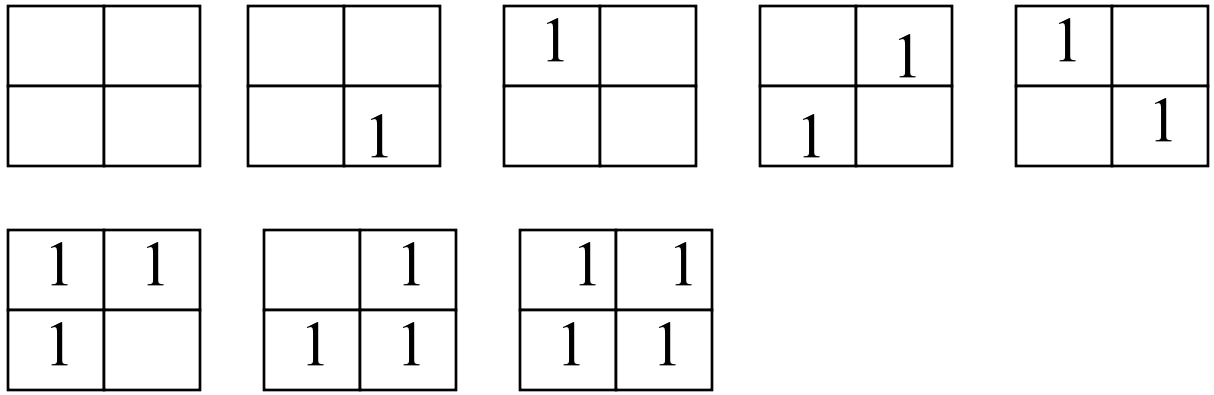
Let us analyze the Positive Davio lattice for 2 variables



Let us analyze all functions that can be created. They are created by assigning values 0 and 1 to coefficients B^i

There are 2*2 * 2 = 8 possible assignments.

- So let us analyze how many symmetric functions of two variables exist.



- It is easy to show that all functions created by the PDavio Lattice are symmetric and their number is the same as for Shannon Lattice.
- **Now by induction one can prove that this is also true for three variables in PDavio Lattice.**

- In another variant one can prove directly by counting functions in PDavio Lattice
- In one more variant one can prove by transforming B^i coefficients to standard S^i coefficient of a symmetric function.
- If you show that every Si coefficient can be transformed to a combination of Bi coefficients then every symmetric function is realizable with Bi coefficients.
- These are simple mathematical exercises that you can do, but the above observations were sufficient to get maximum points.

Problem 8

Mirror image function of any symmetric function is symmetric for same negated variables for which original function is symmetric.

For example :

ab \ cd	00	01	11	10
00				
01			1	
11		1	1	1
10			1	

F=abc+abd+acd+bcd

=

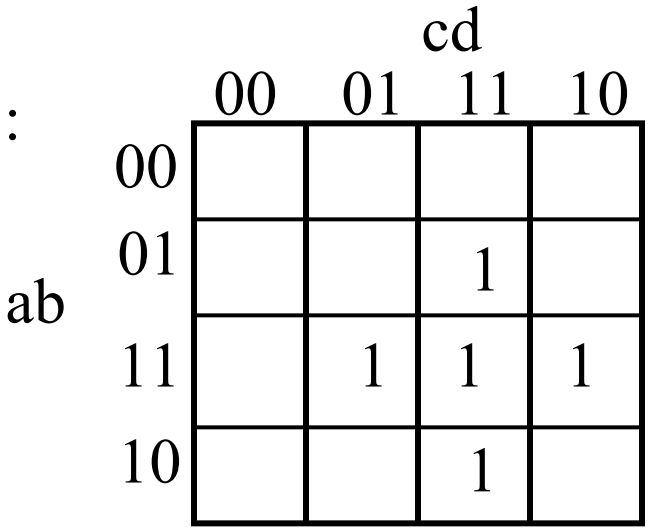
ab \ cd	00	01	11	10
00				
01		1		
11	1	1	1	
10		1		

FG=abc’+abd+c’db+c’da

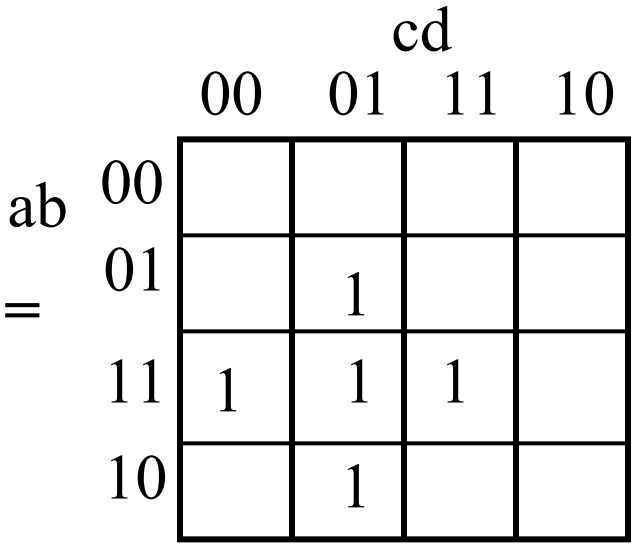
both are symmetric functions

Use this observation to create a more general definition of symmetry and a regular structure to realize all generalized symmetric functions according to your definition. Show solution to function FG, using only multiplexers and negations.

For example :



=



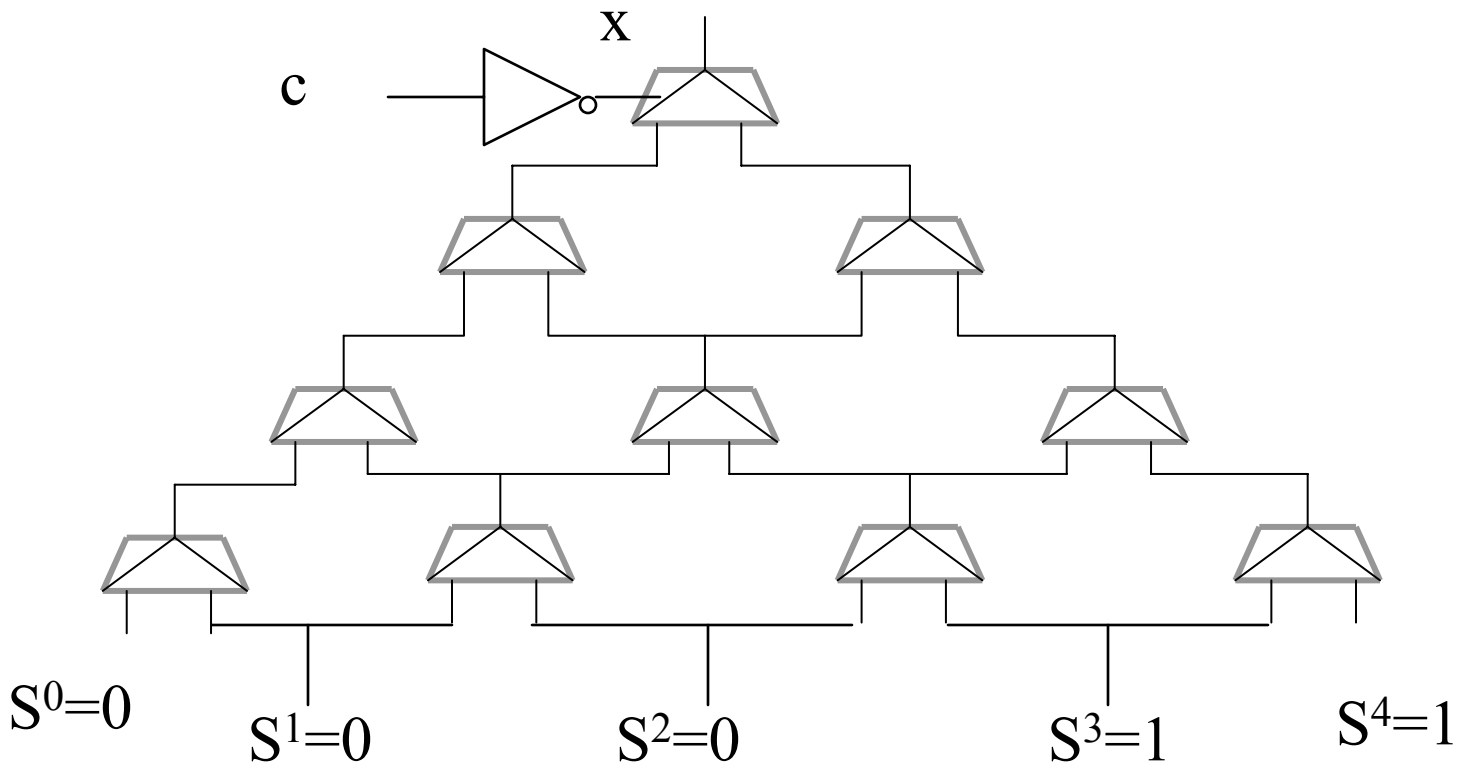
F=abc+abd+acd+bcd

FG=abc'+abd+c'db+c'da

$F(a,b,x,d) = S^{34}(a,b,x,d)$

Observe that the function FG is a symmetric function in a classical sense on variables a, b, x and d if we replace literal c' with new variable x. Thus we can apply any synthesis methods for symmetric function to function FG(a,b,x,d) (which has the same Kmap as function F but the variables in it are a,b,x and d) and next replace the new variable x with the literal c' again. This method can be applied to any regular structure like lattices.

We draw thus a lattice of function F using variables a,b,x and d and next replace variable x with negation of variable c. As we know lattice can be built from only multiplexers, and we add one negation to create c'=x. In fact, instead of adding the negation as a control variable to a multiplexer, it is enough to change the order of data inputs and having the same control input variable c.



A more general definition of symmetry:

A function $F(y_1,y_2,\ldots y_n)$ is symmetric in a generalized sense if for some subset (possible empty or all) of its variables y_i there exist variables x_i being their negations and the function of variables x_i and y_i is symmetric in a classical sense.

Problem 9. removed.

Problem 10. Definitions and ideas.

- 1. Give definitions of prime implicant, prime implicate, essential prime implicant
- 2. Give a definition of a maximum clique of a graph.
- 3. Define Shannon and Davio expansions and show gates for them.
- 4. What the applications of gates and expansions from point 3?
- 5. What are vacuous variables. Give examples.
- 6. Give example of Pareto Minimization.
- 7. Discuss satisfiability versus complementation versus tautology. How you can use a satisfiability solver to check tautology. Show how to do complementation algorithmically using SOP and POS formulas, not Kmaps.

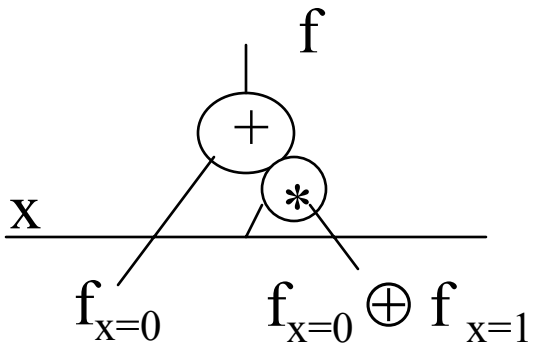
- 1. Give definitions of prime implicant, prime implicate, essential prime implicant
- Implicant g of function f is any function that implies function f. That is, any subset of ones and don't cares in the Kmap of f.
- Product implicant is an implicant that is a product of literals.
- Prime implicant is a product implicant that by removing ANY literal from it, it is no longer an implicant. (if by removing some literal it is still a product implicant than it was not a prime implicant)
- Implicate g of function f is any subset of zeros and don't cares in Kmap of f
- Sum implicate is an implicate that is a sum of literals
- Prime implicate is a product implicate that by removing any literal is no longer an implicate of f
- Essential prime implicant is a prime implicant that it covers certain minterm that is covered only by this prime implicant.

- 2. Give a definition of a maximum clique of a graph.
- Clique of a graph is any set of nodes such that there exist an edge between any two of the nodes from this set.
- Maximum clique of the graph is a clique that is not included in any other clique of the graph

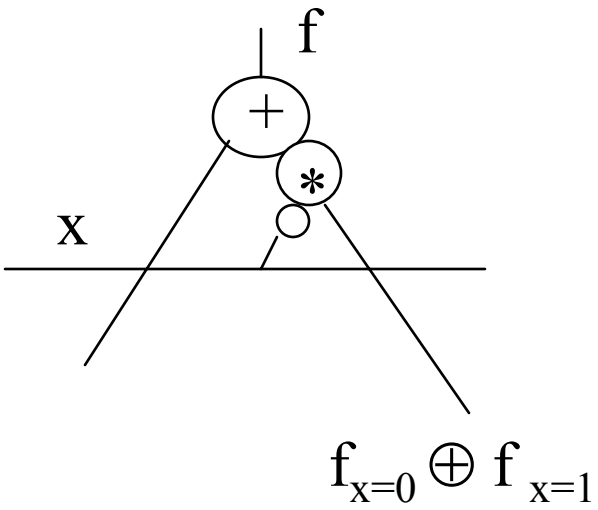
- 3. Define Shannon and Davio expansions and show gates for them.
- Shannon : $f = x' f(x=0) + x f(x=1) = x' f_{x=0} + x f_{x=1} = x'f_0 + x f_1$
- Mux is the gate to realize Shannon expansion.

- Positive Davio : $f = x'f_0 + x f_1 = (1 \oplus x) f_0 \oplus x f_1 = f_0 \oplus x(f_0 \oplus f_1)$

- Negative Davio : $f = f_1 \oplus x'(f_0 \oplus f_1)$



Positive Davio Gate



Negative Davio Gate

- 4. What is the applications of gates and expansions from point 3?
- Multiplexers: multi-level logic synthesis, data path synthesis for arguments of operations and data transfers.
- Positive and Negative Davio – Kronecker Functional Decision Diagrams, Pseudo-Kronecker Decision diagrams, Kronecker Lattices, Reed-Muller Lattices, FPGA – especially Fine Grain FPGAs such as from Concurrent Logic and ATMEL. Generalizations of multiplexers and Shannon expansion in many design methods and theories. Creation of canonical forms and diagrams.
- 5. What are vacuous variables?. Give examples.
- Vacuous are variables that seem to describe the function, but in reality function does not depend on them. Here is an example

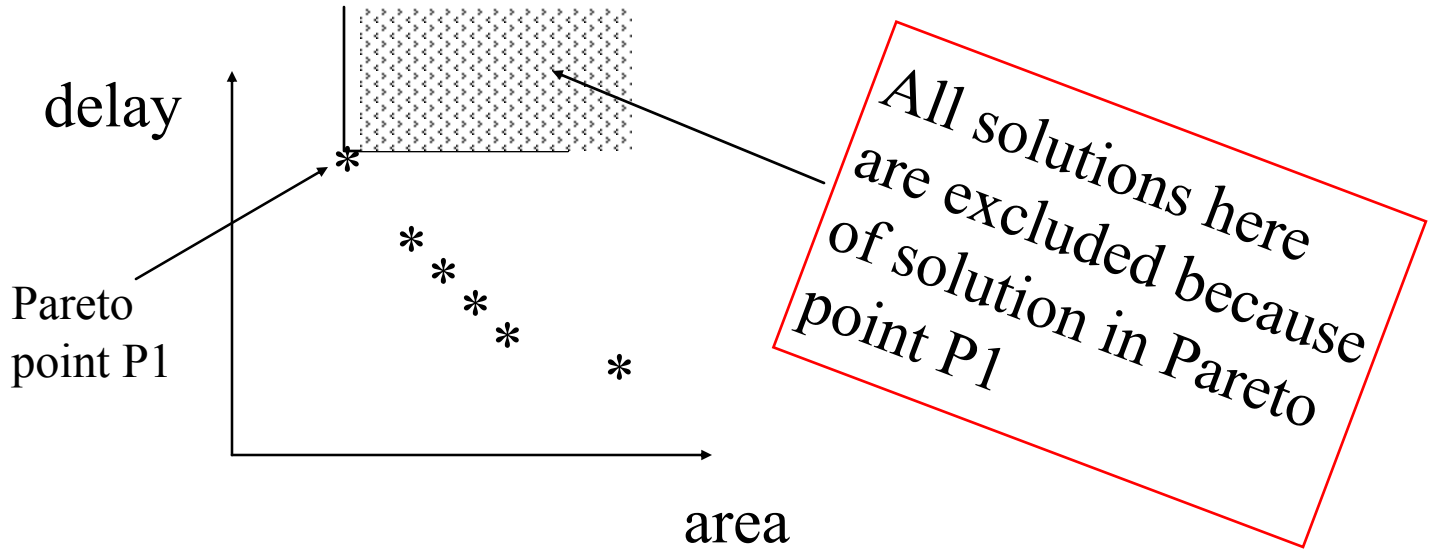
ab \ cd	00	01	11	10
00	0	-	-	0
01	-	-	1	-
11	-	1	-	-
10	0	-	-	0

This function can be minimized to variable b, then variables a,c,d are vacuous (redundant, non-essential).

This function can be minimized to variable d, then a,c,b are vacuous.

Then variables a and c are always vacuous. This can be also verified using the Kmap folding method from class.

- 6. Give example of Pareto Minimization.



- 7. Discuss satisfiability versus complementation versus tautology.

Satisfiability is checking if there exists X that F(X) = 1
(satisfiability is checking if there is a single "1" in Kmap)
Tautology is checking if for all X it holds that F(X) = 1
(tautology is checking if all cells in a Kmap have value of "1")

SAT(F) = NOT { TAUTOLOGY(F') }

A predicate A predicate

Therefore, if you have a SAT solver and complementer you can solve tautology. Similarly if you have a complementer and tautology solver, you can solve SAT.

How you can use a satisfiability solver to check tautology.

SAT(F) = NOT { TAUTOLOGY(F') }

TAUTOLOGY (F) = NOT { SAT (F') }

1. Complement function F to obtain function F'
2. Check if F' is satisfied
3. If F' is satisfied then Tautology (F) is not satisfied.
   And vice versa.

Show how to do complementation algorithmically using SOP and POS formulas, not Kmaps.

I am showing on an example, using de Morgan twice.

(ab' + b'c'd' + c e)' = (ab')' (b'c'd')' (ce)' = (a'+b) (b+c+d) (c'+e')

Since the result should be in SOP form again, you need to multiply or use the branching method to find all products of literals that satisfy the right side.

Problem 11. Algorithms.

- 1. List all known to you methods to solve the covering problem.
- 2. What is the difference of Binate and Unate Covering
- 3. Boolean equations and Petrick Functions. Show examples.
- 4. Color the graph from Figure A and prove that this coloring is minimal
- 5. Assuming that this graph is a compatibility graph, draw the incompatibility graph and its clique covering.

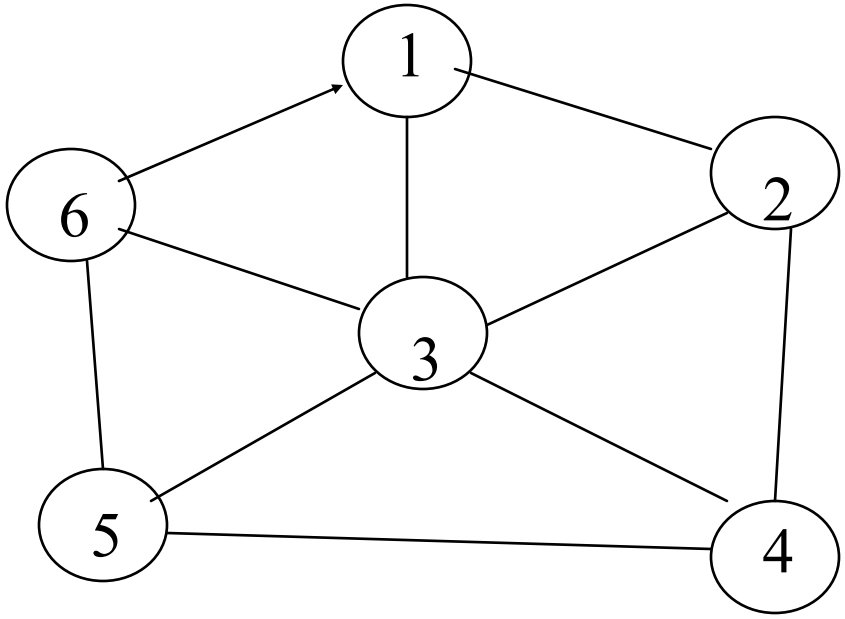


Figure A

- **1. List all known to you methods to solve the covering problem.**
- METHOD 1. Based on Covering Table (Covering Matrix) write the Petrick Function and solve it using a branching method, integer programming method, Boolean multiplication method or any of the methods similar to modern SAT solvers.
- METHOD 2. Based on Covering Table, directly find solution to it using Backtracking or any search algorithm such as A*, Depth First, Breadth First etc

- **2. What is the difference of Binate and Unate Covering**
- In Unate covering the Decision Function (called Petrick Function) is a PRODUCT of variables in positive polarity. Because the Decision function is Positive Polarity (Unate) the problem is called Unate Covering.

- 3. Boolean equations and Petrick Functions. Show examples.
- Method of using Boolean Equations comes from Boole himself as an adaptation of Decartes Method (Cartesius). It can be simplified this way:
- A. Reduce the problem to a set of sub-problems – formulate an equation for each sub-problem.
- B. Convert every equation for a sub-problem to a Boolean equation. For instance, you can convert it to a sum of products, Product of sums, or sum of literals. But sometimes these equations can still have implication operators (like in AND/OR trees) or EXOR operators (like in Helliwell function).
- C. Make a Boolean Product of all the equations. Sometimes you could try to convert the final formula (decision function) to a POS. If yes, you can use standard SAT solvers that are very efficient. If converting to POS would create too many variables or terms, you have to check satisfiability on your formula as it is. This makes the method less efficient, but still very useful to solve many logical problems.

- Petrick function for Covering Table illustrated by an example.
  For each column create a sum term. Make product for all columns.

	1	2	3	4	5	6	7
A	1	0	0	1	1	0	0
B	0	1	0	0	1	1	0
C	1	1	0	1	0	0	1
D	0	0	1	1	0	0	1
E	1	0	0	1	0	1	1

PETRICK FUNCTION

= 1 =

(A+C+E)(B+C)(D)(A+C+D+E)(A+B)

(B+E)(C+D+E)

(A+C+E)

Row A or Row C or Row E must be taken to cover column 1

(B+C)

Row B or Row C must be taken to cover column 2

- 4. Color the graph from Figure A and prove that this coloring is minimal
- 5. Assuming that this graph is a compatibility graph, draw the incompatibility graph and its clique covering.

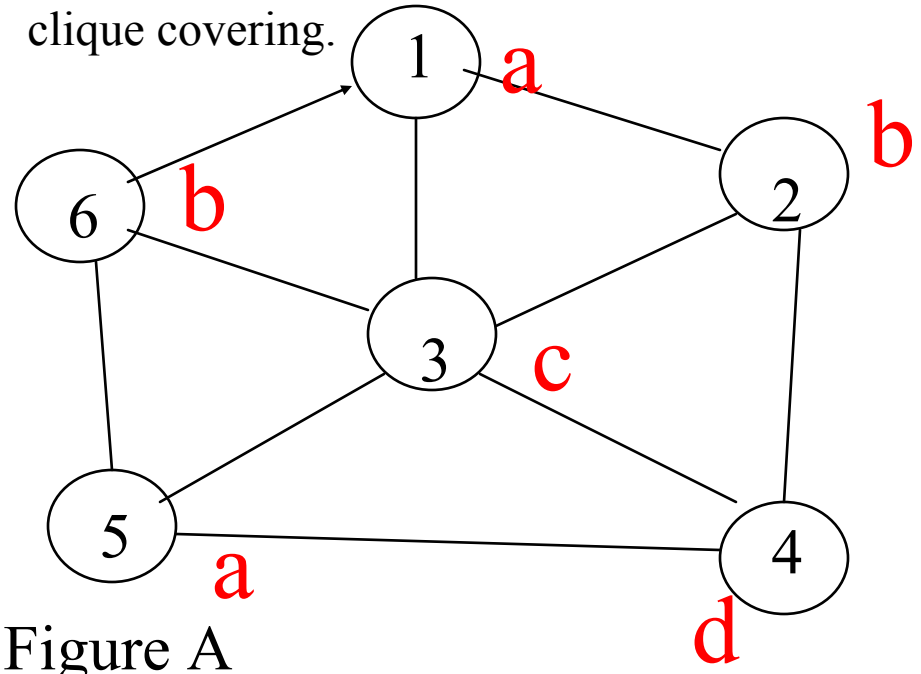
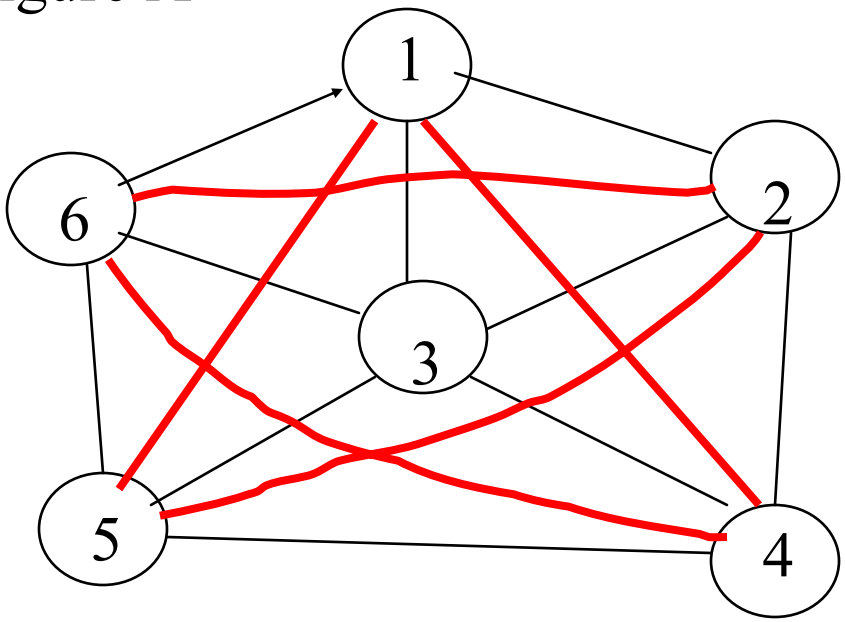
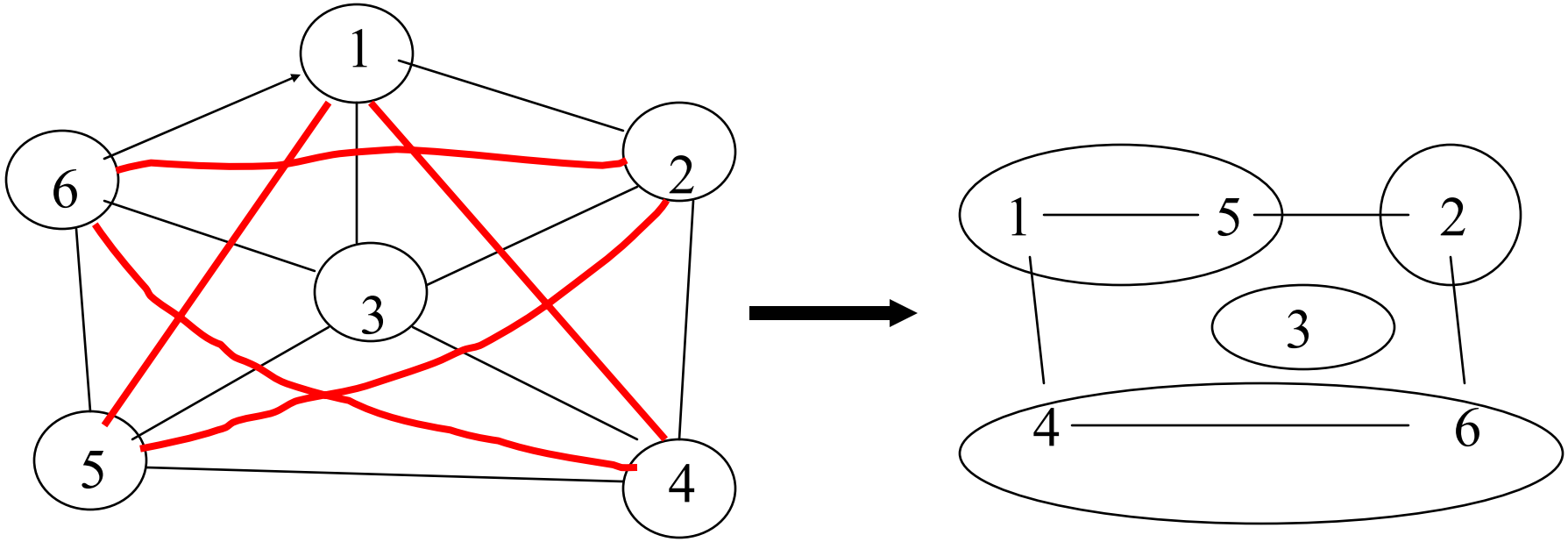


Figure A



1. Since nodes 1,2,3 are a clique, then the coloring as shown in red is the only one possible with accuracy to isomorphism.
2. Now node 6 can be colored with b
3. Now node 5 can be colored with a.
4. Now node 4 cannot be colored with a, cannot be colored with b and cannot be colored with d.
5. So 4 colors is minimum. It can be also proven formally by backtracking and illustrating the backtracking problem by a graph as was done in class.

Incompatibility graph is shown in red, this is a complement graph to compatibility graph. The union of both graphs is a full graph.



- It is obvious that all other clique partitionings will be isomorphic to the one shown above.
- Thus the minimum number of independent sets of incompatibility graph is four
- Thus the minimal number of colors (chromatic number) of the compatibility graph is four, as we found earlier.

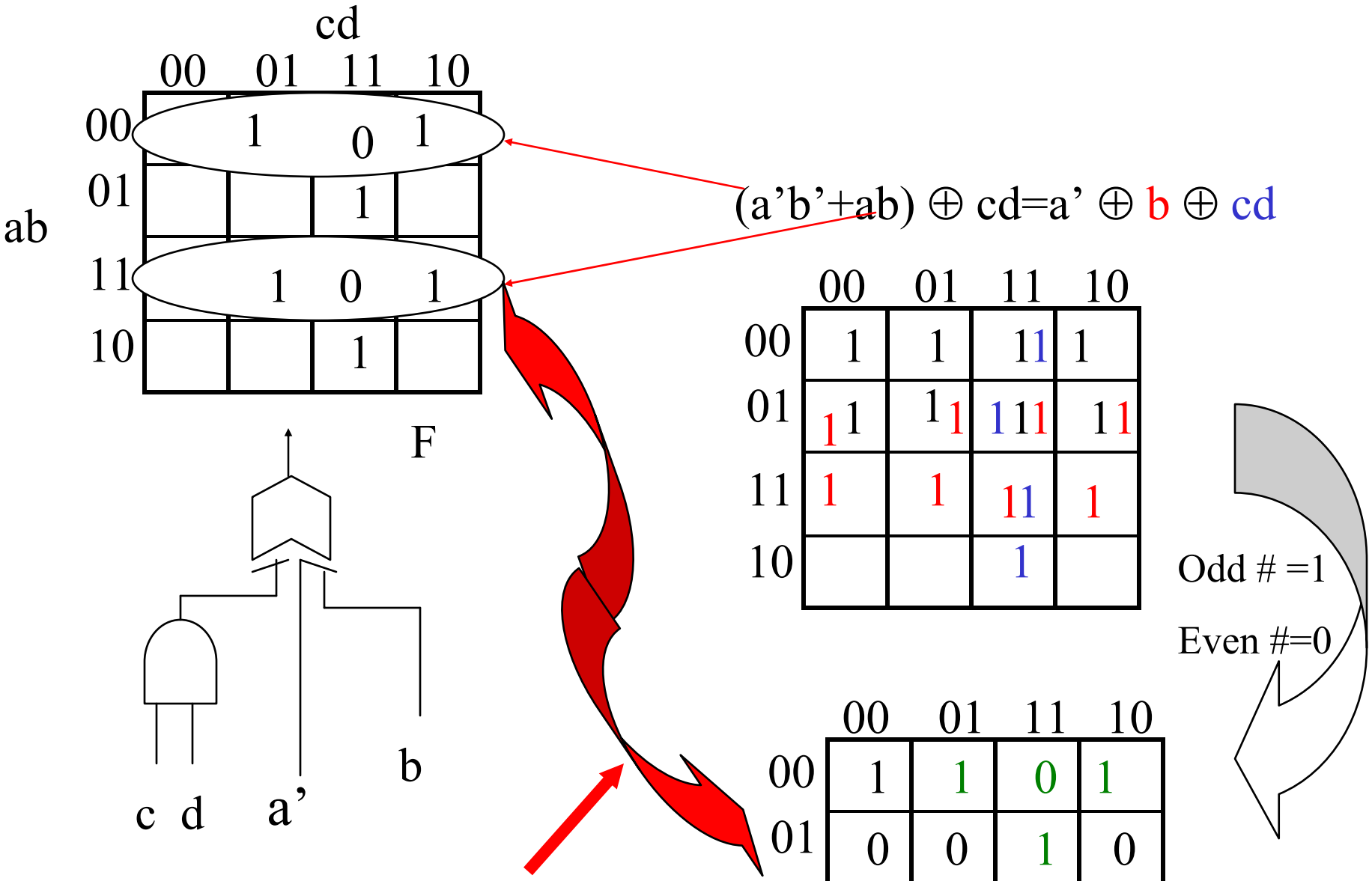
Problem 12

1. Find the minimal ESOP circuit.
2. Draw its schematic
3. Verify graphically that your solution is correct.

Use the concept of incomplete tautology.

ab \ cd	00	01	11	10
00		1	0	1
01			1	
11		1	0	1
10			1	

All other cells are don't cares


cd
00 01 11 10
ab
00
01
11
10
1 0 1
1
1 0 1
1
F
(a'b'+ab) ⊕ cd=a' ⊕ b ⊕ cd
00 01 11 10
00 1 1 11 1
01 11 11 111 11
11 1 1 11 1
10 1
Odd # =1
Even #=0
00 01 11 10
00 1 1 0 1
01 0 0 1 0
11 1 1 0 1
10 0 0 1 0
G
c d a' b
Verification. Those with green color are the same. So functions F and G are an incomplete tautology